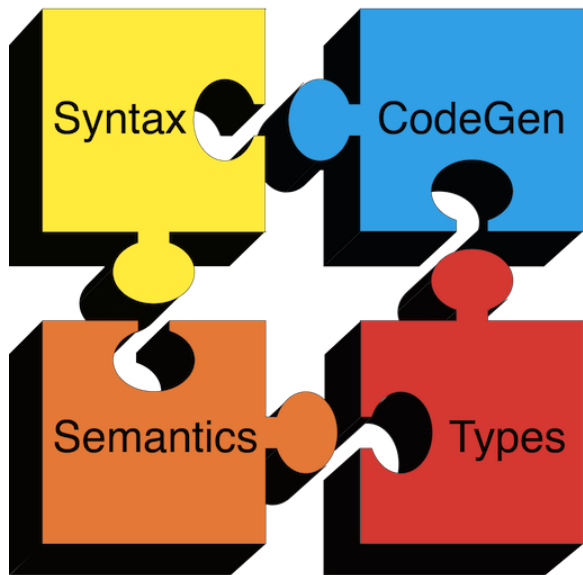




Compilers

Object Layout

- OO implementation = Basic code generation + More stuff
- OO Slogan: If B is a subclass of A, then an object of class B can be used wherever an object of class A is expected
- This means that code in class A works unmodified for an object of class B

- How are objects represented in memory?
- How is dynamic dispatch implemented?

Object Layout

```
Class A {  
  a: Int <- 0;  
  d: Int <- 1;  
  f(): Int { a <- a + d };  
};
```

```
Class B inherits A {  
  b: Int <- 2;  
  f(): Int { a };  
  g(): Int { a <- a - b };  
};
```

```
Class C inherits A {  
  c: Int <- 3;  
  h(): Int { a <- a * c };  
};
```

Object Layout

```
Class A {  
  a: Int <- 0;  
  d: Int <- 1;  
  f(): Int { a <- a + d };  
};
```

```
Class B inherits A {  
  b: Int <- 2;  
  f(): Int { a };  
  g(): Int { a <- a - b };  
};
```

```
Class C inherits A {  
  c: Int <- 3;  
  h(): Int { a <- a * c };  
};
```

Attributes **a** and **d** are inherited by
classes **B** and **C**

Object Layout

```
Class A {  
  a: Int <- 0;  
  d: Int <- 1;  
  f(): Int { a <- a + d };  
};
```

```
Class B inherits A {  
  b: Int <- 2;  
  f(): Int { a };  
  g(): Int { a <- a - b };  
};
```

```
Class C inherits A {  
  c: Int <- 3;  
  h(): Int { a <- a * c };  
};
```

All methods in all classes refer to **a**

Object Layout

```
Class A {  
  a: Int <- 0;  
  d: Int <- 1;  
  f(): Int { a <- a + d };  
};
```

```
Class B inherits A {  
  b: Int <- 2;  
  f(): Int { a };  
  g(): Int { a <- a - b };  
};
```

```
Class C inherits A {  
  c: Int <- 3;  
  h(): Int { a <- a * c };  
};
```

For **A** methods to work correctly in **A**, **B**, and **C** objects, attribute **a** must be in the same “place” in each object

- Objects are laid out in contiguous memory
- Each attribute stored at a fixed offset in the object
 - The attribute is in the same place in every object of that class
- When a method is invoked, the object is **self** and the fields are the object's attributes

- The first 3 words of Cool objects contain header information:

	Offset
Class Tag	0
Object Size	4
Dispatch Ptr	8
Attribute 1	12
Attribute 2	16
...	

- Class tag is an integer
 - Identifies class of the object
- Object size is an integer
 - Size of the object in words
- Dispatch ptr is a pointer to a table of methods
 - More later
- Attributes in subsequent slots
- Lay out in contiguous memory

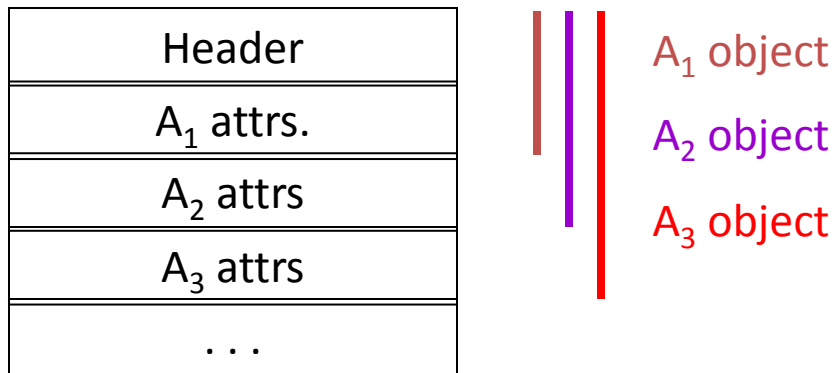
Observation: Given a layout for class **A**, a layout for subclass **B** can be defined by extending the layout of **A** with additional slots for the additional attributes of **B**

Leaves the layout of **A** unchanged
(**B** is an extension)

Object Layout

Offset Class	0	4	8	12	16	20
A	Atag	5	*	a	d	
B	Btag	6	*	a	d	b
C	Ctag	6	*	a	d	c

- The offset for an attribute is the same in a class and all of its subclasses
 - Any method for an A_1 can be used on a subclass A_2
- Consider layout for $A_n < \dots < A_3 < A_2 < A_1$



For the given classes and object layout table, what are the correct inheritance relationships between classes?

- ☐ $A < B < C$
- ☐ $C < B < A$
- ☐ $A < C < B$
- ☐ $B < C < A$

Object Layout

```
Class A inherits ??? {  
  u: Int <- 0;  
  v: Int <- 1;  
};
```

```
Class B inherits ??? {  
  x: Int <- 3;  
  y: Int <- 4;  
};
```

```
Class C inherits ??? {  
  z: Int <- 5;  
};
```

Class Tag
Object Size
Dispatch Ptr
x
y
z
u
v
...

Object Layout

```
Class A {  
  a: Int <- 0;  
  d: Int <- 1;  
  f(): Int { a <- a + d };  
};
```

```
Class B inherits A {  
  b: Int <- 2;  
  f(): Int { a };  
  g(): Int { a <- a - b };  
};
```

```
Class C inherits A {  
  c: Int <- 3;  
  h(): Int { a <- a * c };  
};
```

Consider the dispatch
e.g()

Object Layout

```
Class A {  
  a: Int <- 0;  
  d: Int <- 1;  
  f(): Int { a <- a + d };  
};
```

```
Class B inherits A {  
  b: Int <- 2;  
  f(): Int { a };  
  g(): Int { a <- a - b };  
};
```

```
Class C inherits A {  
  c: Int <- 3;  
  h(): Int { a <- a * c };  
};
```

Consider the dispatch
e.f ()

- Every class has a fixed set of methods
 - including inherited methods
- *A dispatch table* indexes these methods
 - An array of method entry points
 - A method **f** lives at a fixed offset in the dispatch table for a class and all of its subclasses

Object Layout

Offset	0	4
Class		
A	fA	
B	fB	g
C	fA	h

- The dispatch table for class **A** has only 1 method
- The tables for **B** and **C** extend the table for **A** to the right
- Because methods can be overridden, the method for **f** is not the same in every class, but is always at the same offset

- The dispatch pointer in an object of class X points to the dispatch table for class X
- Every method f of class X is assigned an offset O_f in the dispatch table at compile time

- To implement a dynamic dispatch $e.f()$ we
 - Evaluate e , giving an object x
 - Call $D[O_f]$
 - D is the dispatch table for x
 - In the call, $self$ is bound to x